# Types of Data Storage

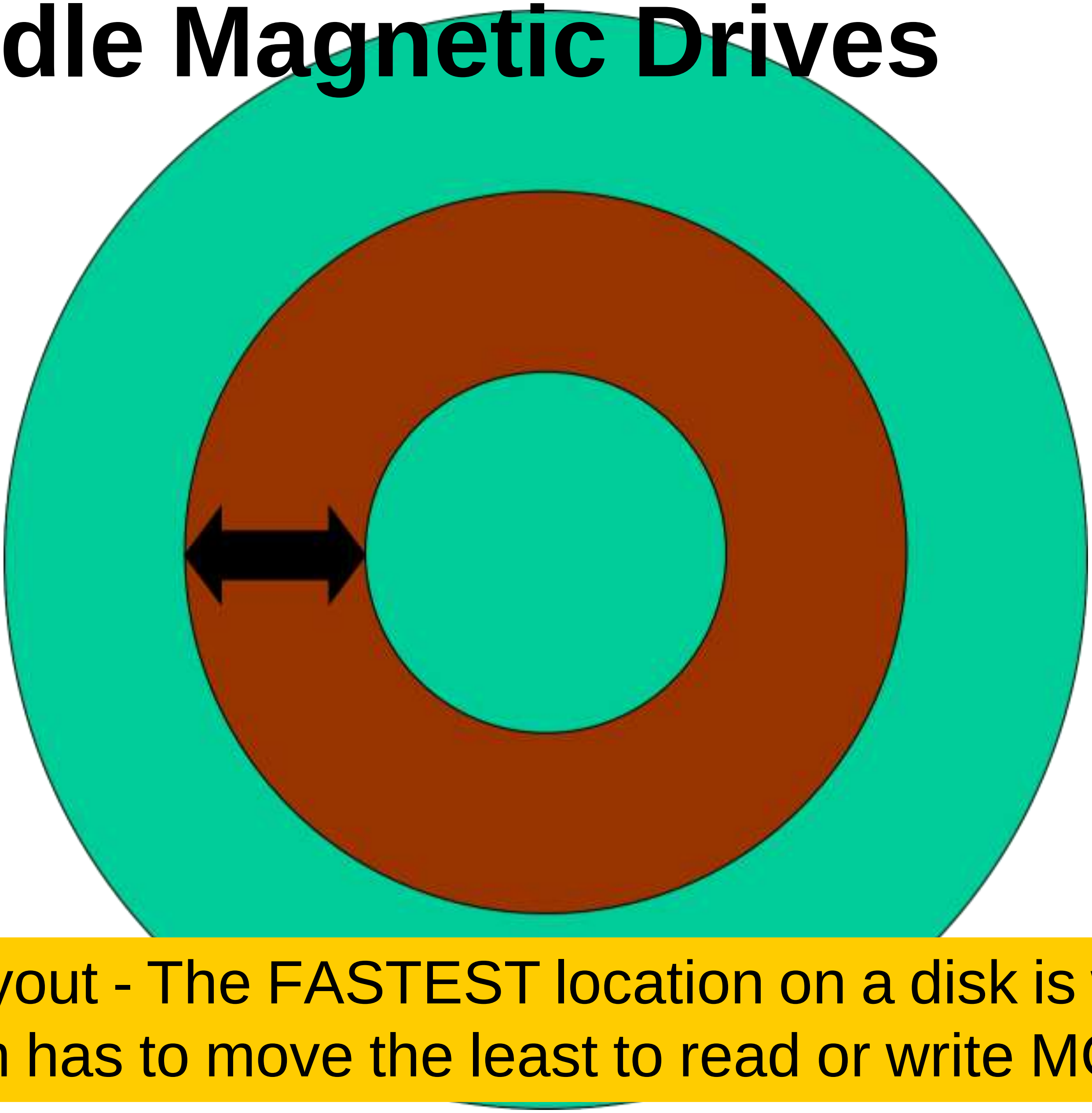
- SSD Drives
- Spindle Magnetic Drives
  - High Speed - 15,000 RPM spindle speed for high access rate data (600GB)
  - 10,000 RPM spindle speed drives for historical data (1 TB)
  - Common Disk Drives – 7,200 and 5,400 spindle speed drives ( 1-8 TB)
- Storage Area Network (SAN)

# Spindle Magnetic Drives

Disk is organized into sectors.
The disk arm moves to a spot
to read a byte of data

**Western Digital VelociRaptor 300 GB,Internal,10000 RPM,3.5" (WD3000BLFS) Hard Drive**

# Spindle Magnetic Drives



Disk Layout - The FASTEST location on a disk is where the disk arm has to move the least to read or write MOST data

# Solid State Disk (SSD)

A **solid-state drive** (**SSD**) is a nonvolatile storage device that stores persistent data on **solid-state** flash memory. **Solid-state drives** actually aren't **hard drives** in the traditional sense of the term, as there are no moving parts involved

Disk is organized into cells.
Each byte is directly addressable and readable.

# Solid State Disk (SSD)

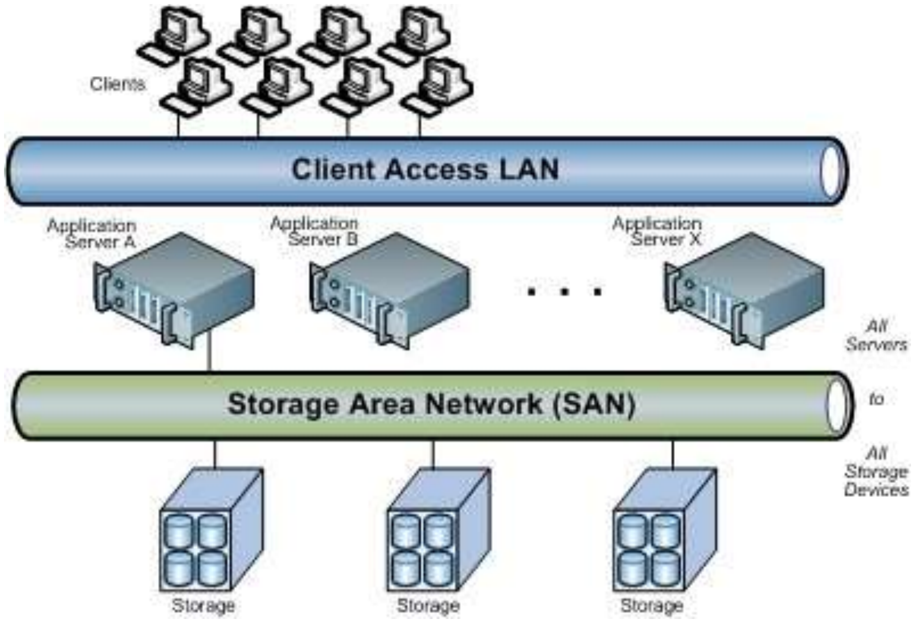
- Performance varies widely by type and manufacture (larger size is faster)
- Drive lifespan and reliability is improving
- Minimum 3 times faster than a hard disk on the same machine
- New SSD drive with M.2 interface will be much faster…

# Fusion Drives

- Combine SSD drives as a cache with a slower spindle disk drive
- Most used data is in the cache
- Works great until your active data set exceeds the size of the cache
- Requires CPU cycles to sync cache with disk drive

# Storage Area Network (SAN)

- High-speed network that provides block-level access to storage
- Composed of hosts, switches, storage elements, and storage devices that are interconnected

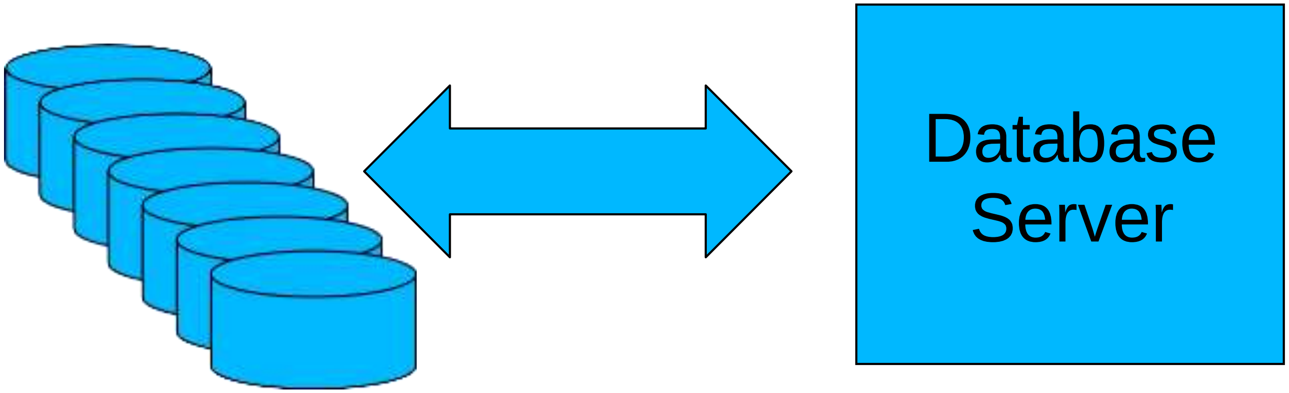


# Disk Interface

- SCSI – Small Computer System Interface
- SATA – Serial ATA (3 GB/s – 6 GB/s)
- SAS – Serial Attached SCSI (6 GB/s – 12 GB/s)
- Fiber Channel – High-speed network technology (32-128 GB/s)
- Thunderbolt – Developed by Intel and Apple (10 GB/s – 40 GB/s)

# Best Performance

- Multiple Disks attached to Fast Interfaces



# What is RAID?

- Redundant Array of Independent Disks
- Goal is to increase:
  - Performance
  - Reliability
  - Safety of Data

# RAID 1 - Safest

- Exact copy of two or more disks
- Safe – if one disk fails the other will continue
- Fast – reads can be twice as fast (reads from both disks at the same time)
- The array will continue to operate as long as at least one member drive is operational

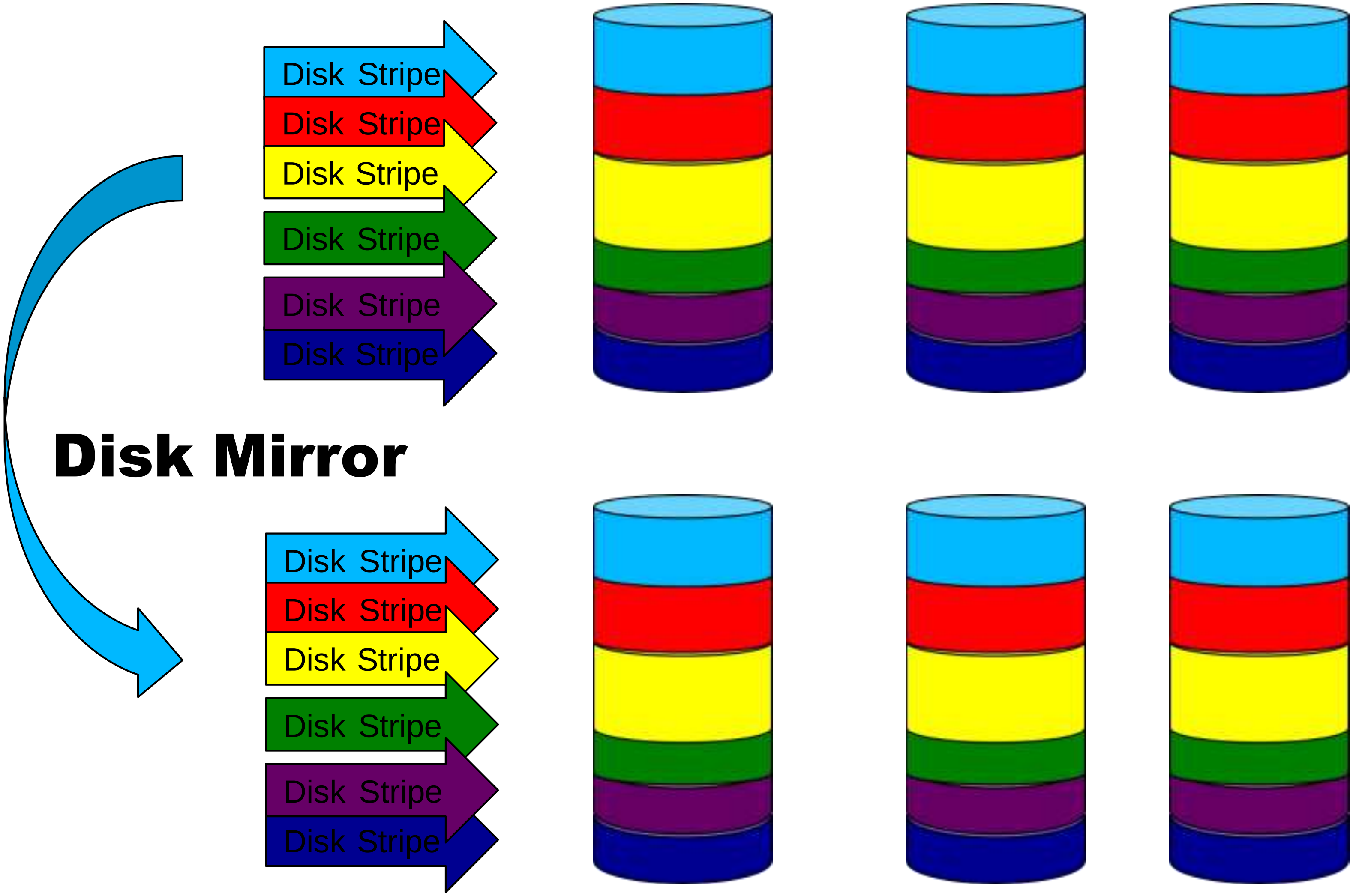
# RAID 0 - Fast

- Stripes data evenly across two or more disks
- No redundancy and no fault-tolerance
- Performance – provides data at rates up to ‘n’ times faster where ‘n’ is the number of disks

# RAID 10 – Safest + Fastest

- Combines the Mirroring of RAID 1 with the Striping of RAID 0
- Best Performance
- Best Reliability and Safety
- Best Configuration is a smaller Block/Strip Size
  - Recommend 64K or 128K

# RAID 10 Configuration



# RAID 10 Configuration

- Recommended Stripe/Block Size for Databases with a 2K to 16K Page Size
  - 64K  best
  - 128 K
  - 256 K is the max Size
- Filesystems benefit from a larger Stripe/Block Size – NOT Databases

# RAID 5, 6, 7… etc. - Dangerous

- **RAID 5** is block-level striping with distributed parity. It requires that all drives except one be present to operate.
- Upon failure of a single drive, reads can be calculated from the distributed parity such that no data is lost.
- ***Parity Calculation is slow and may fail***

# Do NOT Use RAID 6,5, .. etc

- Why is RAID5 more popular than RAID10 among storage administrators?
  - It requires fewer drives to deliver the same storage capacity.
  - Storage salespeople can present a less expensive proposal to meet the required storage volume. They make a slanted case to make a sale!
  - Most people have never studied the issue themselves and so trust that RAID5 is good.

# CERN Disk Drive Failure Rates

- Testing at CERN after they experienced data loss on RAID5 arrays determined:
  - Most drive failures (80%) are caused by hardware and firmware failure (another 10% from wrong firmware version).
  - Partial media failure accounts for much of the rest of the data loss experienced on both magnetic and SSD drives as they aged.
  - They experienced cosmic ray damage flipping bits, equally on both magnetic and SSD type drives.

# CERN Disk Drive Failure Rates

- Drives today are commodity priced. Is the risk of data loss worth the relatively small savings?
- The failure rates and expected lifespans for “premium” drives are identical to commodity retail drives. So, it doesn’t help that you are spending >$1000 per drive.
- According to a storage industry study, failure of a second drive is 4X more likely than the single drive failure rate would predict!
- Atomic writes across multiple drives in a RAID5 array are not guaranteed!

# CERN Disk Drive Failure Rates

- Larger drives take longer to rebuild increasing the likelihood of losing a second drive.
- A recent study concluded that drives over 1TB are statistically likely to suffer from unrecoverable multiple bit dropouts.
- The number of bits on the drive exceeds the bit failure rate!
- The error rates as observed by the CERN study on silent corruption are far higher than the official rate of one in every 10^16 bits.
- The observed error rate was about one in 10^7 bits, or 1 out of about 1 in every 1,000,000 bits (~125,000 bytes).

# RAW vs Cooked Space

- RAW – Informix has direct access to the device and space
  - Used to be 25% faster
  - Support for RAW devices is dwindling
  - Harder to manage
- Cooked – Informix accesses the UNIX Filesystem to access a space

# RAW Space

- RAW – Informix has direct access to the device
- No UNIX OS Overhead
- No UNIX OS Buffering
- No UNIX Management of devices
- Used to be 25% faster
- Support for RAW devices is dwindling

# Using Raw Devices

- When most UNIX systems create a device, they will create two means of accessing that device.
- Block mode of access. This is used by the UNIX file system. The device will have a name that does not begin with the letter “r” and a permissions display in “ls” that begin with the letter “b”. The following is an example:

```
brw-------          1 sysinfo  sysinfo    1, 15 Jun 21 1995 /dev/dsk/0s1
```

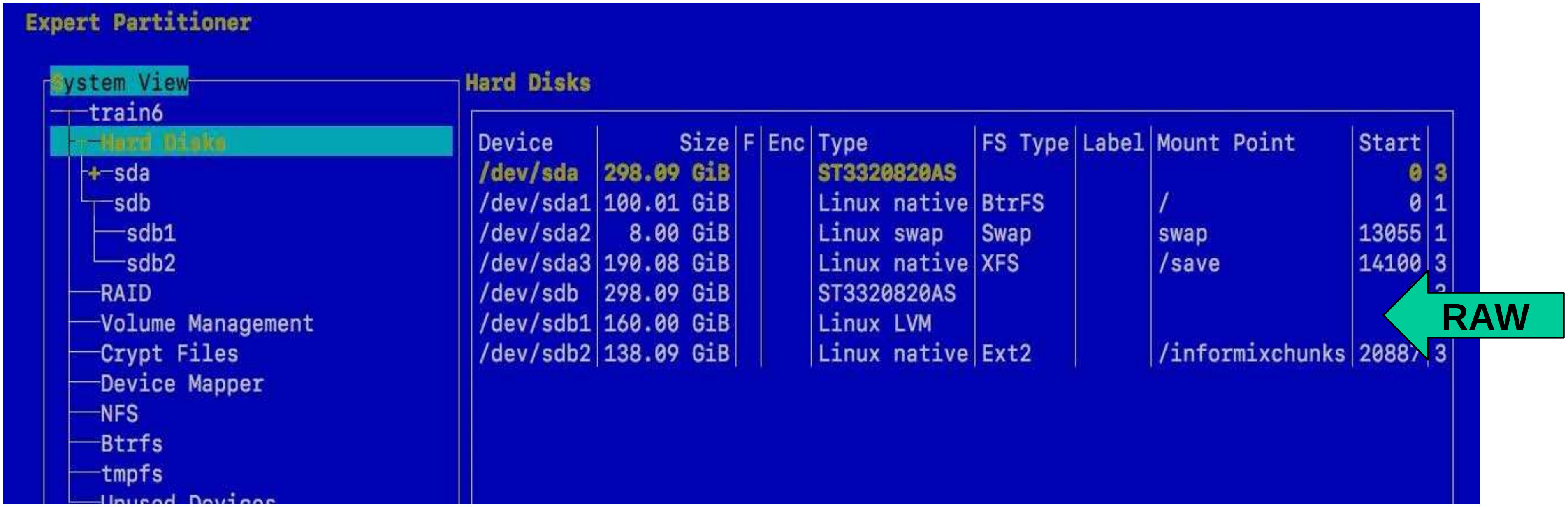
- ***Raw mode of access. The device will have a name that begins with “r” and the permission displays will begin with the letter “c”. The following is an example:***

```
crw-------          1 sysinfo  sysinfo    1, 15 Jun 21 1995 /dev/rdsk/0s1
```

**Always use *raw mode device* for your Informix chunks.**

# Using Raw Devices

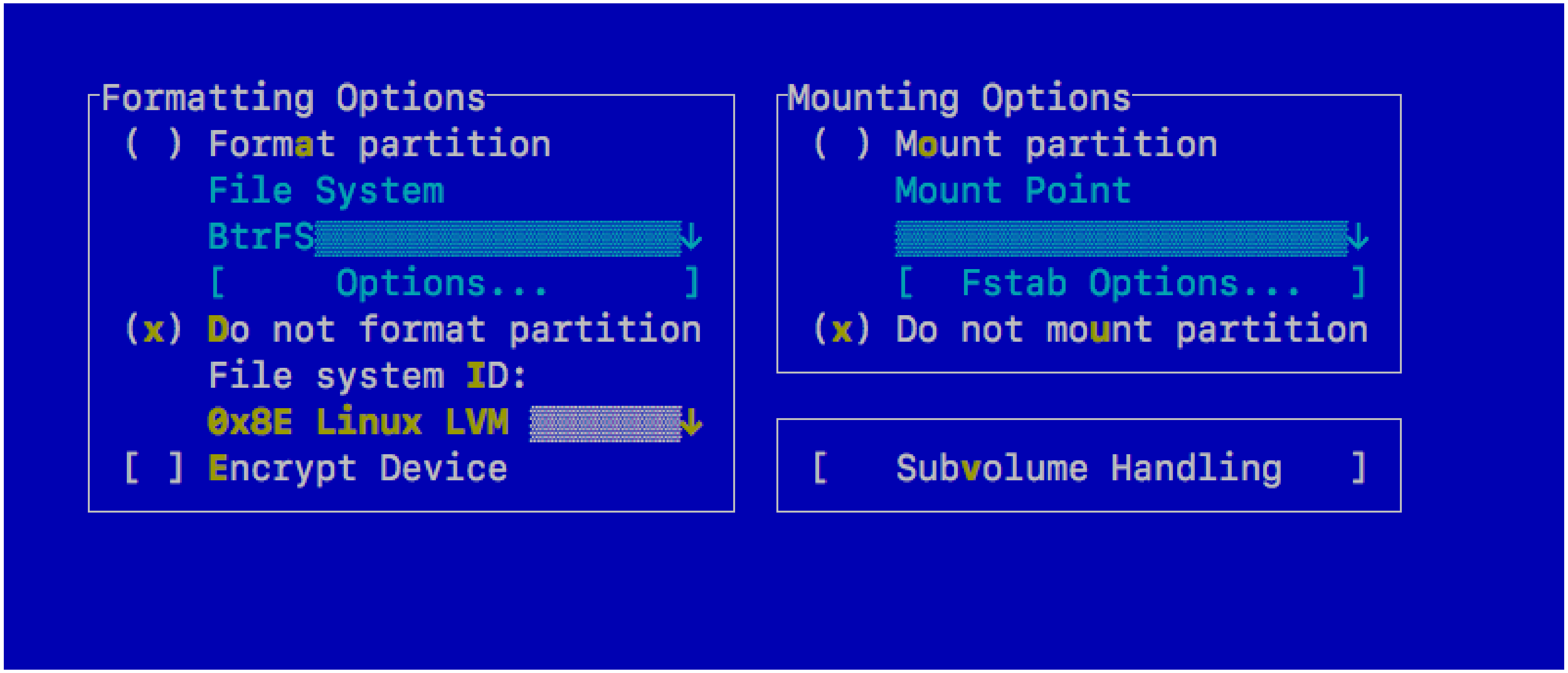
- Create the Raw device with the disk partition utility



```
Expert Partitioner

System View                     Hard Disks
──train6
  ├─Hard Disks                  Device    │    Size │F│Enc│Type         │FS Type│Label│Mount Point     │Start │
  │ ├+─sda                      /dev/sda  │298.09 GiB│ │   │ST3320820AS  │       │     │                │     0│3
  │ └─sdb                       /dev/sda1 │100.01 GiB│ │   │Linux native │BtrFS  │     │/               │     0│1
  │   ├─sdb1                    /dev/sda2 │  8.00 GiB│ │   │Linux swap   │Swap   │     │swap            │ 13055│1
  │   └─sdb2                    /dev/sda3 │190.08 GiB│ │   │Linux native │XFS    │     │/save           │ 14100│3
  ├─RAID                        /dev/sdb  │298.09 GiB│ │   │ST3320820AS  │       │     │                │      │3
  ├─Volume Management           /dev/sdb1 │160.00 GiB│ │   │Linux LVM    │       │     │                │      │
  ├─Crypt Files                 /dev/sdb2 │138.09 GiB│ │   │Linux native │Ext2   │     │/informixchunks │ 20887│3
  ├─Device Mapper
  ├─NFS
  ├─Btrfs
  ├─tmpfs
  └─Unused Devices
```

# Using Raw Devices

- Do NOT Format
- Do NOT Mount



```
┌Formatting Options──────────────────┐  ┌Mounting Options───────────────────┐
│ ( ) Format partition               │  │ ( ) Mount partition               │
│     File System                    │  │     Mount Point                   │
│     BtrFS░░░░░░░░░░░░░░░░░░░░░↓    │  │     ░░░░░░░░░░░░░░░░░░░░░░░░░↓    │
│     [      Options...        ]     │  │     [   Fstab Options...   ]      │
│ (x) Do not format partition        │  │ (x) Do not mount partition        │
│     File system ID:                │  └───────────────────────────────────┘
│     0x8E Linux LVM ░░░░░░░↓        │  ┌───────────────────────────────────┐
│ [ ] Encrypt Device                 │  │ [    Subvolume Handling      ]    │
└────────────────────────────────────┘  └───────────────────────────────────┘
```

# Using Raw Devices

- Review your OS for specific steps to create Raw Devices

# Cooked Space

- Cooked – All Informix access to the device is through the OS
- OS Overhead and OS Buffering
- Simple Management of Devices
- May be Slower
- Pick your filesystem carefully – do not use a Journaled Filesystem

# Using Cooked Devices

- Avoid Journaling Filesystems
  - Worst - EXT4, EXT3 (with journaling enabled), ZFS, BTFS
  - Acceptable - JFS2/OpenJFS
- Best - On Linux: EXT2 or EXT3 with journaling disabled

# Creating Cooked Files

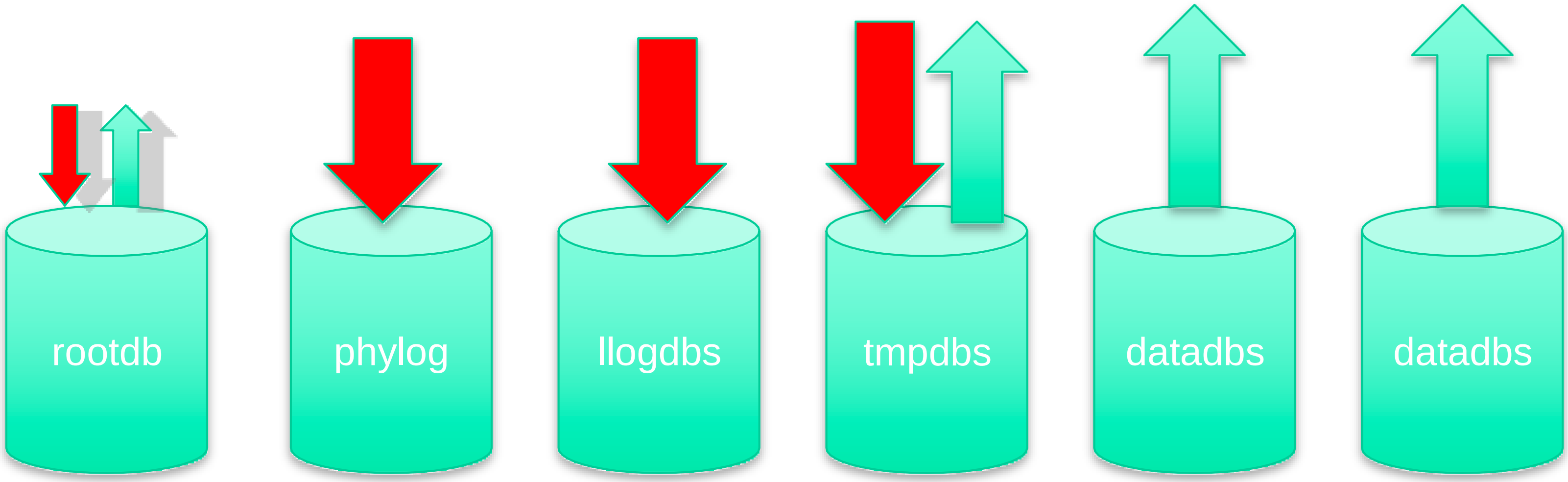
- Create empty file with touch
  - touch $INFORMIXCHUNKS1/datadbs
- Change the permissions for Informix
  - chmod 660 $INFORMIXCHUNKS1/datadbs
- Change the owner to informix and group to informix
  - chown informix:informix $INFORMIXCHUNKS1/datadbs
- Create links if necessary
  - ln -s $INFORMIXCHUNKS1/datadbs  $INFORMIXLINKS/datadbs

# Dbspaces, Chunks, and Pages

- Dbspace
- Chunk
- Page
- Extent
- Tablespace
- Partition
- Fragment

# Database Disk I/O

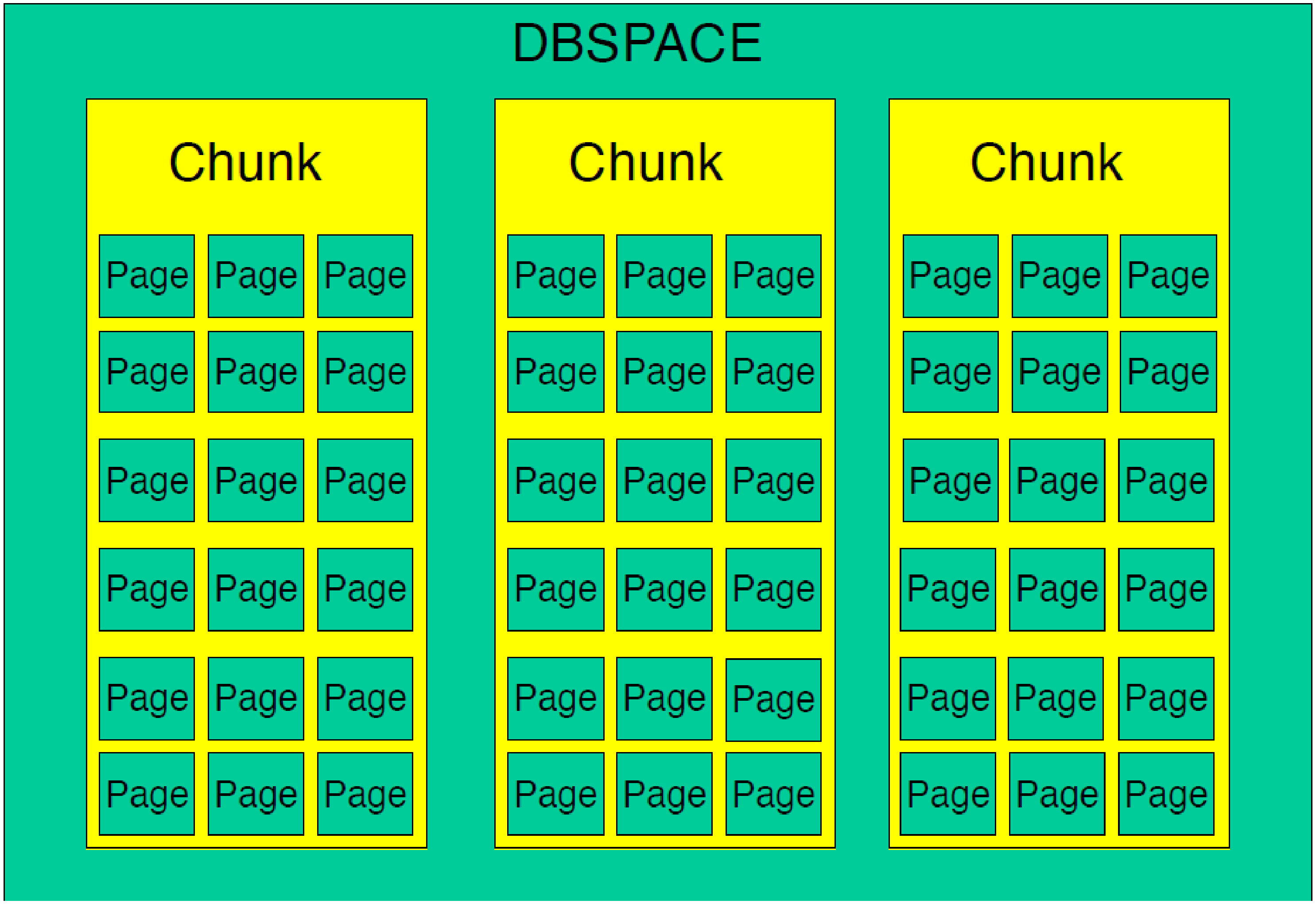
- Most Reads are from Data and Tables
- Writes will be split between Physical Log, Logical Log, Temp, and Data



# Managing Disk Space

- Page – A page is the <u>physical unit of disk storage</u> that Informix uses to read from and write to Informix databases. The size of a page varies from computer to computer and from dbspace to dbspace. The default page holds either 2 or 4 kilobytes. Because the rootdb, logical log and physical log dbspaces must use the default page size. Since Informix v10.00 the page size for other dbspaces is configurable in multiples of the server's default page size from the default up to 16K.
- Chunk – Chunk is the <u>unit of physical disk</u> dedicated to Informix data storage. It represents an allocation of cooked file, cooked device, or raw disk space and is the only unit of physical storage that the Informix administrator allocates.
- DBspace – A dbspace is a <u>logical container of one or more chunks</u>.

# Disk Layout - Pages, Chunks, and Dbspaces



DBSPACE

Chunk

Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page

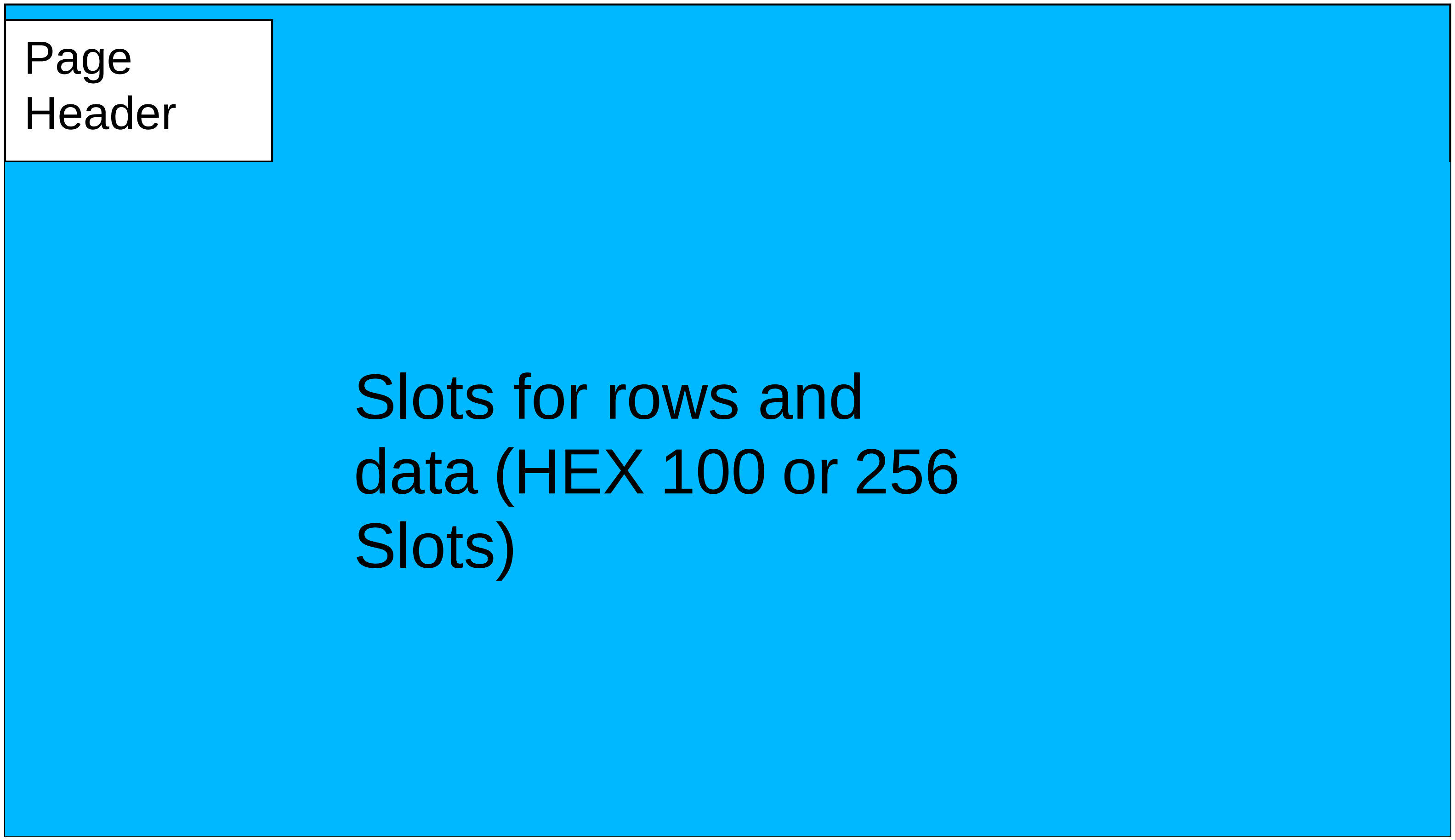
Chunk

Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page

Chunk

Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page
Page Page Page

# Page Layout



Page Header

Slots for rows and data (HEX 100 or 256 Slots)

# Dbspace Considerations

- What is the optimal page size for my data? 2K, 4K, 8K, or 16K?
- Partition table extent sizing (onspaces -ef & -en)
- Should the dbspace be expandable (automate adding chunks from the storage pool)?

# Types of Dbspace

- “Normal” dbspaces
- Temporary dbspaces
- Blobspaces
- Unlogged Smart Blobspaces
- Logged Smart Blobspaces
- Physical Log Dbspace

# Chunk Considerations

- Should the chunk be extendable?
- How active will the data be?
- What else is stored on the device?
- Where should it be placed within storage?
  - SSD
  - Fast magnetic disk
  - Slower magnetic disk
- Should the chunk have an Informix mirror chunk?

# Root DBspace

- Define in your ONCONFIG File using the base page size of your Informix port
- Should be on your fastest storage
- Nothing should be in your Root DBspace except
  - Reserved Pages
  - Sysmaster Database
  - Sysutilites Database
  - Sysusers Database

# Root DBspace

- Move out of the Root DBSpace
  - Physical Logs
  - Logical Logs
  - Sysadmin Database
  - Temp Dbspace

# Physical Log DBspace

- The Physical and Logical log will have 30-50% of all writes
- Move out of Root to separate Dbspaces
- Physical Log Size = 1.25 x Buffer Size
- A Checkpoint will occur when the Physical Log is 75% Full

# Logical Log DBspace

- The Physical and Logical log will have 30-50% of all writes
- Move out of Root to separate Dbspaces
- Logical Log Size = Hold 5-10 minutes of transactions at peak time
- Have enough Logical Logs for 4 days

# Temp DBspace

- Up to 50% of all I/O can be to the Temp DBspaces (reads + writes)
- Create at least 3 non-logged Temp DBspaces (min) – Informix will use them in round robin and in parallel
- Create 1-3 Logged Temp DBspaces for Logged Temp Transactions if needed

# Index DBspace

- Indexes are optimal on 16K page Dbspaces
- Create 16K page size DBSpace for indexes
- Create a 16K BUFFERPOOL for indexes

# Data DBspace

- Create Data Dbspaces based on the row size of your tables.
- How may rows will fit on a page?
- What is the least amount of wasted space?
- Spread your data across multiple DBspaces.

# How to calculate Optimal Dbspace Size for a Table?

| Table Name | Row Size | Rows 2 K page | Waste on 2 K page | Rows 4 K page | Waste on 4 K page | Rows 8 K page | Waste on 8 K page | Rows 16 K page | Waste on 16 K page |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

- DBspace Page Size (2 K, 4 K, 8 K, 16 K) – 28 bytes = Size for Data ( 2048-28=2020)
- Row Size / Dbspace Size for Data = Number of Rows per page
- Waste is how much space is left over
- See Art Kagel’s script named “waste” or Lester’s Sysmaster script

# Partition Large Tables

- Any table larger than a typical chunk may benefit by being partitioned into multiple partitions or tablespaces
- Tables with more than 16,777,216 pages must be partitioned into multiple partitions or fragments
- Partition tables for performance

# Informix Storage Pool

- The storage pool contains:
  - the directories, cooked files, and raw devices
  - May be used to automatically expand an existing dbspace
- Storage space threshold is defined in the SP_THRESHOLD parameter
- Database scheduler will automatically run a task that expands the space, either by extending an existing chunk in the space or by adding a new chunk
- Configured in sysadmin database, storagepool table

# Informix Storage Pool

- Large Predefined Files to be used to automate expanding DBspaces as needed.
- Current configuration allows only a single storage pool
- Disadvantage: Some applications may require multiple storage pools to allow isolating DBspaces at the storage level. Example:
  - Transaction tables versus historical tables
  - Data tables versus indexes versus blobspace versus temp space

# Creating an Informix Storage Pool

- Use the Sysadmin task or admin function to create a Storage Pool
  - EXECUTE FUNCTION task("storagepool add", "path", "begin_offset", "total_size", "chunk size", "priority");
- Specify the following:
  - The path for the file, directory, or device to use when additional storage space is required
  - The offset in KB into the device to begin allocating space ( 0 for cooked files)
  - The total space available to Informix for this device
  - The minimum size in KB of a chunk that can be allocated from the device (1000 KB or greater)
  - The priority of the device (1 = high; 2 = medium; 3 = low) for space allocation

# Extendable Chunks

- Since version 11.70 any chunk can be marked “extendable”. When such a chunk fills, the engine will automatically extend the file size a specified amount.
- Only valid for cooked filesystem chunks
- Attribute set after chunk creation with an API function:

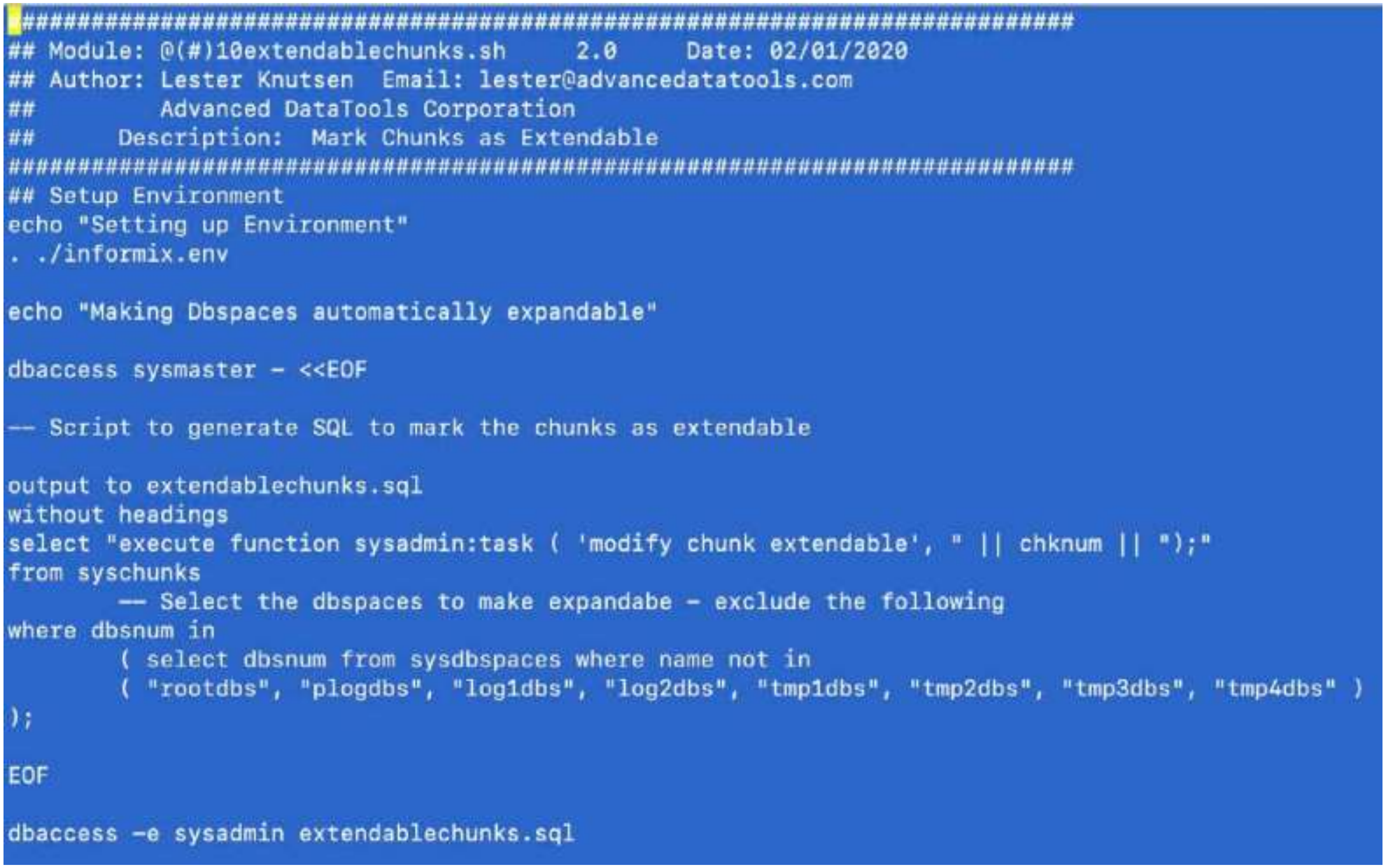
```
   EXECUTE FUNCTION sysadmin:task("modify chunk extendable", "4");
```

- Extension size is set by DBspace in an API function:

```
   EXECUTE FUNCTION task("modify space sp_sizes", "DBspace",
"new_sz_kb", "min_extend_sz", "max_size");
```

Sets the initial size of new chunks created by dbspace expansion, the maximum size of a chunk for this dbspace, and the amount to extend extendable chunks in this dbspace.

# Extendable Chunks Script



```
#############################################################################
## Module: @(#)10extendablechunks.sh     2.0     Date: 02/01/2020
## Author: Lester Knutsen  Email: lester@advancedatatools.com
##         Advanced DataTools Corporation
##      Description:  Mark Chunks as Extendable
#############################################################################
## Setup Environment
echo "Setting up Environment"
. ./informix.env

echo "Making Dbspaces automatically expandable"

dbaccess sysmaster - <<EOF

-- Script to generate SQL to mark the chunks as extendable

output to extendablechunks.sql
without headings
select "execute function sysadmin:task ( 'modify chunk extendable', " || chknum || ");"
from syschunks
        -- Select the dbspaces to make expandabe - exclude the following
where dbsnum in
        ( select dbsnum from sysdbspaces where name not in
        ( "rootdbs", "plogdbs", "log1dbs", "log2dbs", "tmp1dbs", "tmp2dbs", "tmp3dbs", "tmp4dbs" )
);

EOF

dbaccess -e sysadmin extendablechunks.sql
```

# Simplify Disk Management

- Dbspaces can be configured to expand by adding new chunks when the existing chunks have been used up.
- Create a storage pool of disk files and devices to use to expand dbspaces.
- Mark dbspaces as expandable.
- If no chunks in those dbspaces are extendable or are extendable but have reached maximum size, the engine will allocate a new chunk from the storage pool.

# Disk Layout Best Practices

- Use Symbolic Links
- Avoid Disk IO Contention
- Mirror Critical Media
- Move the Logical and Physical Logs
- Create 3 or more Temp DBspaces
- Isolate High-Use Tables
- Separate Indexes from Data

# Use Symbolic Links

- Once a device has been associated with a chunk in Informix it cannot be easily changed (a server restore is required).
- This makes it very difficult to change device names if you need to change disk drives.
- **Recommendation**: use the UNIX facility to create symbolic links and use ONLY symbolic links as the chunk paths passed to onspaces.

<u>Example:</u>

```
ln -s /dev/rds1s1 /usr/informix/dev/server1/chunk1
```

If the physical disk /dev/rsd1s1 needs to be replaced with a new disk, it may not have the same device name. By using symbolic links, you can create the link to the new device and easily copy the data from /dev/rds1s1 (using dd) or restore Informix from archive.

# Avoid Disk IO Contention

- Goals for efficient disk layout typical in a production environment:
  - Limiting disk head movement
  - Reducing disk contention
  - Balancing the load
  - Maximizing availability
- You must make some trade-offs between these goals when you design your disk layout. For example, separating the system catalog tables, the logical logs, and the physical log on physically separate devices can help reduce contention for these resources; however, this action can also increase the chances that you have to perform a system restore.

# Mirror Critical Media

- Mirror the critical media – If your chunks are not built from mirrored disk pairs, the root DBspace, the DBspace containing the physical log, and the DBspace containing the logical log files should be mirrored using Informix mirroring. You must specify mirroring on a chunk-by-chunk basis. Locate the primary and the mirrored chunk on different disks. Ideally, different controllers should handle the different disks.
- Mirror speeds up Disk Reads – Informix will read different sectors from both the primary and the mirror at the same time, increasing throughput.

# Move the Logical and Physical Logs from the Rootdbs

- The logical log and physical log both contain data that Informix writes frequently. Likewise, reserved pages are read frequently; they contain internal tables that describe and track all DBspaces, blobspaces, chunks, databases, and tblspaces.
- By default, Informix stores the logical and physical logs together in the root DBspace when a new server is initialized. Maintaining these together in the root DBspace will become a source of contention as your database system grows.
- Reduce this contention and provide better load balancing by moving the logical and physical logs to separate partitions or, even better, separate disk drives. For optimum performance, create two or more additional DBspaces:
  - one for the physical log (take advantage of the new physical log DBspace type v12.10+)
  - one or more for the logical log (if more than one alternate creating logs round robin).
- When you move the logs, avoid storing them in a db space/disk that contains high-access rate tables; instead consider storing them in a DBspace dedicated to storing only the physical or logical log.

# Create 3 or More Temp DBspaces

- Informix will read/write multiple temp DBspaces in parallel and create temp tables fragmented across them.
- Sort-work files are written round robin to all temp DBspaces.
- Merging sort-work files will read from two temp DBspaces and write to a third if available.
- Temp DBspaces may have high levels of activity
- Move temp DBspaces to separate disks
- ONCONFIG file

```
DBSPACETEMP tmp1dbs:tmp2dbs:tmp3dbs:normal_dbspc
```

# Isolate High-Use Tables

- Place a table with high I/O activity on a disk device dedicated to its use and thus reduce contention.
- Put the tables partitions with the highest frequency of use on the fastest drives.
- Placing two high-access tables on separate disk devices reduces competition for disk access when joins are formed between the two tables or when the two tables experience frequent, simultaneous access from multiple applications.

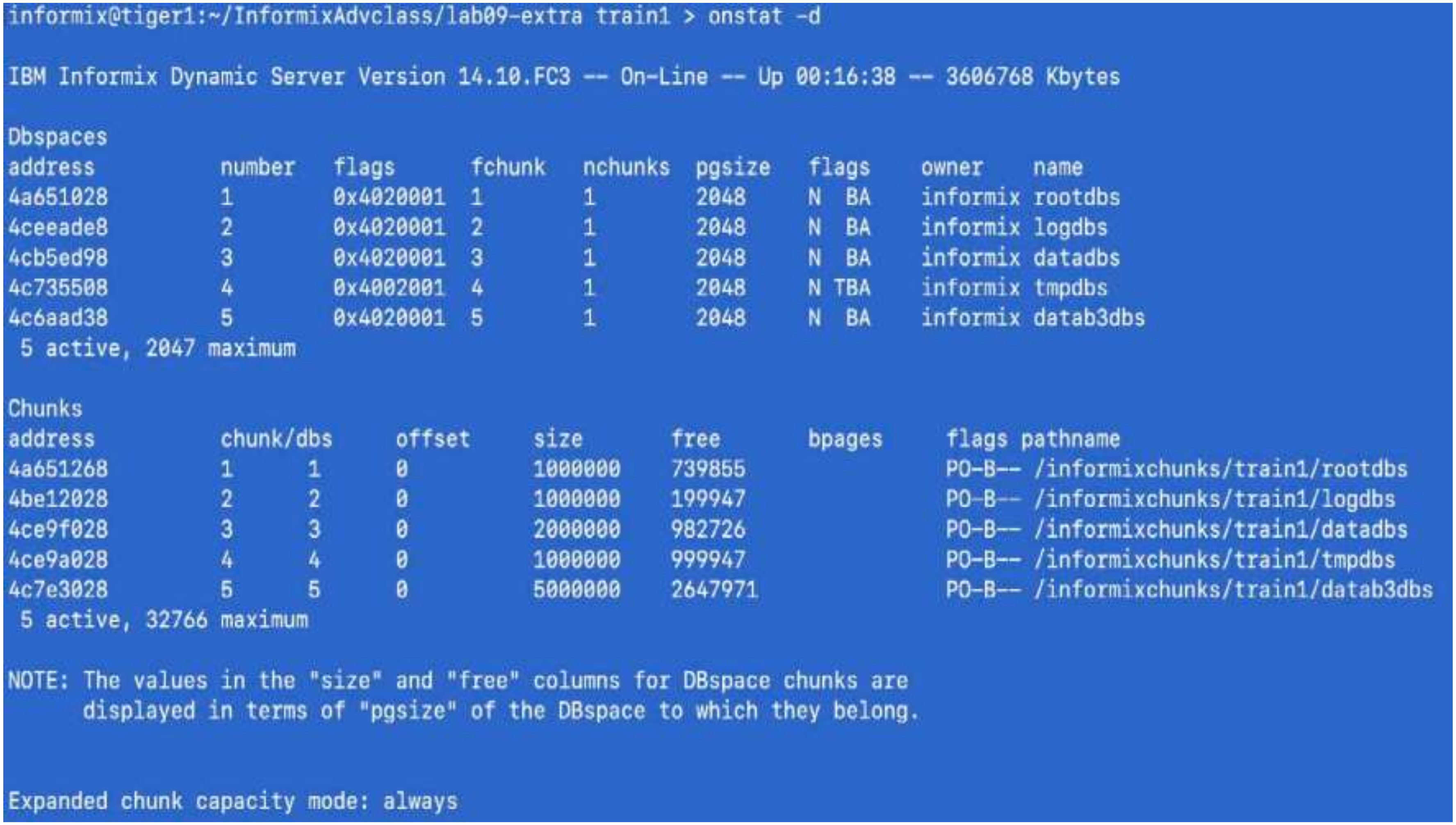
# Separate Indexes from Data

- Create a DBspace for data
- Create a separate DBspace for indexes
- Informix will read indexes and data in parallel

# Monitoring Disk Performance

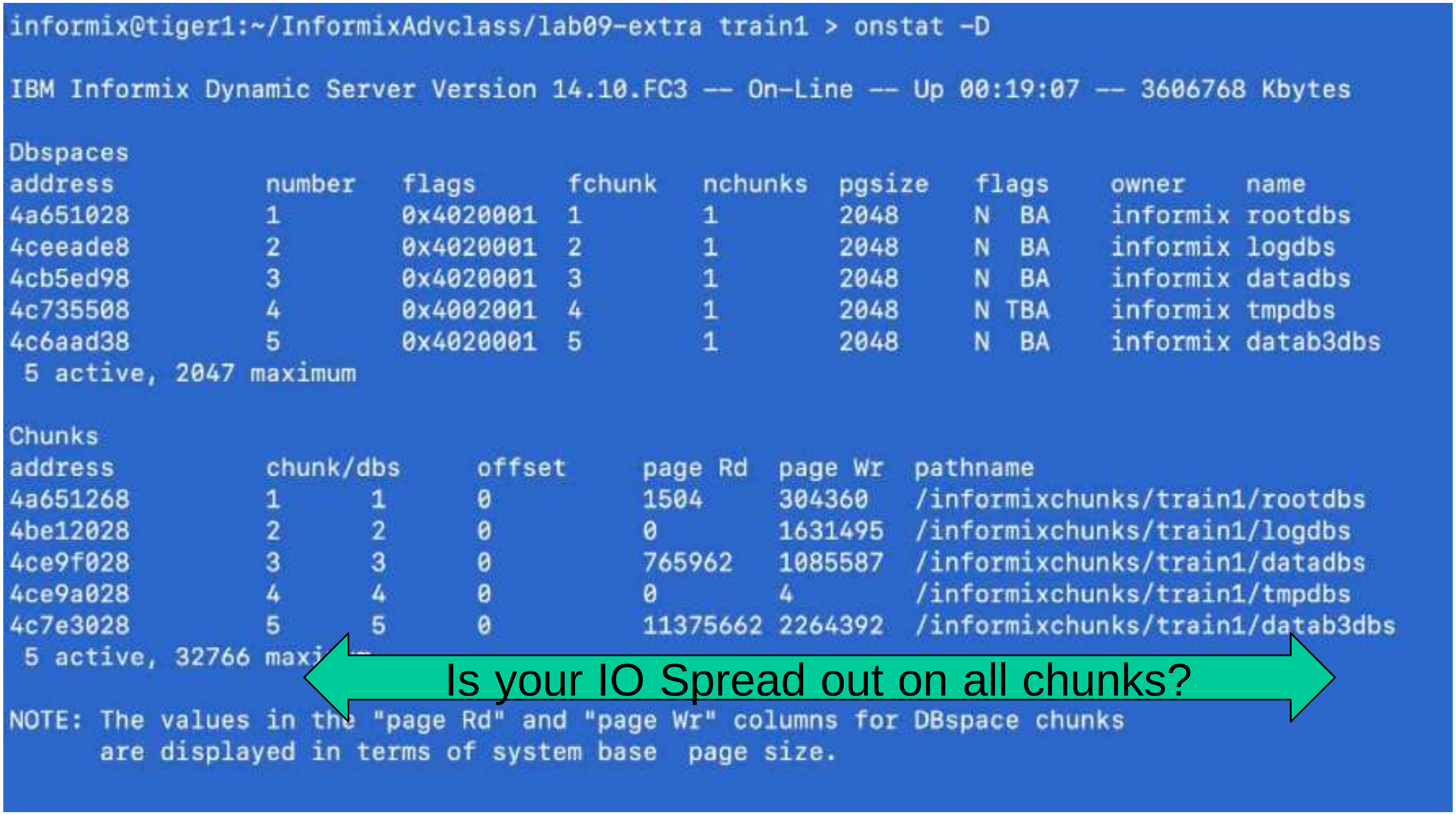
| Onstat Command | Description |
|---|---|
| onstat -d | Print dbspaces and chunks |
| onstat -D | Print dbspaces and chunk IO |
| onstat -g iof | Print disk IO statistics by chunk/file |
| onstat -g iov | Print disk IO statistics by vp |
| onstat –g ioh | Print IO history for the last hour by chunk |

# Monitor with Onstat -d



```
informix@tiger1:~/InformixAdvclass/lab09-extra train1 > onstat -d

IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:16:38 -- 3606768 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
4a651028         1        0x4020001  1        1        2048     N  BA    informix rootdbs
4ceeade8         2        0x4020001  2        1        2048     N  BA    informix logdbs
4cb5ed98         3        0x4020001  3        1        2048     N  BA    informix datadbs
4c735508         4        0x4002001  4        1        2048     N TBA    informix tmpdbs
4c6aad38         5        0x4020001  5        1        2048     N  BA    informix datab3dbs
 5 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
4a651268         1      1      0          1000000    739855                PO-B-- /informixchunks/train1/rootdbs
4be12028         2      2      0          1000000    199947                PO-B-- /informixchunks/train1/logdbs
4ce9f028         3      3      0          2000000    982726                PO-B-- /informixchunks/train1/datadbs
4ce9a028         4      4      0          1000000    999947                PO-B-- /informixchunks/train1/tmpdbs
4c7e3028         5      5      0          5000000    2647971               PO-B-- /informixchunks/train1/datab3dbs
 5 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

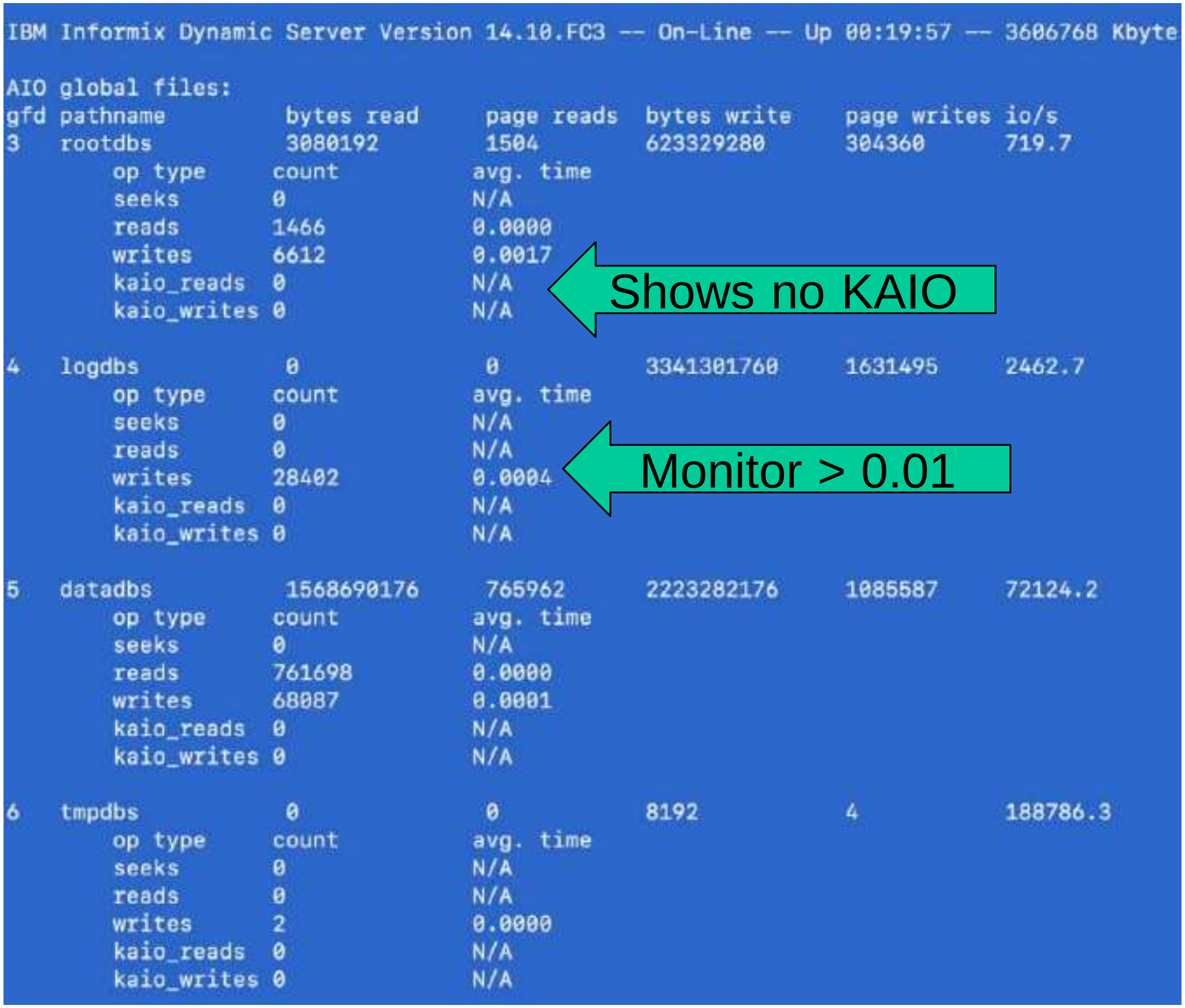
# Monitor with Onstat -D



```
informix@tiger1:~/InformixAdvclass/lab09-extra train1 > onstat -D

IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:19:07 -- 3606768 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
4a651028         1        0x4020001  1        1        2048     N  BA    informix rootdbs
4ceeade8         2        0x4020001  2        1        2048     N  BA    informix logdbs
4cb5ed98         3        0x4020001  3        1        2048     N  BA    informix datadbs
4c735508         4        0x4002001  4        1        2048     N TBA    informix tmpdbs
4c6aad38         5        0x4020001  5        1        2048     N  BA    informix datab3dbs
 5 active, 2047 maximum

Chunks
address          chunk/dbs     offset     page Rd  page Wr  pathname
4a651268         1      1      0          1504     304360   /informixchunks/train1/rootdbs
4be12028         2      2      0          0        1631495  /informixchunks/train1/logdbs
4ce9f028         3      3      0          765962   1085587  /informixchunks/train1/datadbs
4ce9a028         4      4      0          0        4        /informixchunks/train1/tmpdbs
4c7e3028         5      5      0          11375662 2264392  /informixchunks/train1/datab3dbs
 5 active, 32766 maximum

NOTE: The values in the "page Rd" and "page Wr" columns for DBspace chunks
      are displayed in terms of system base  page size.
```

Is your IO Spread out on all chunks?

# Monitor with onstat –g iof

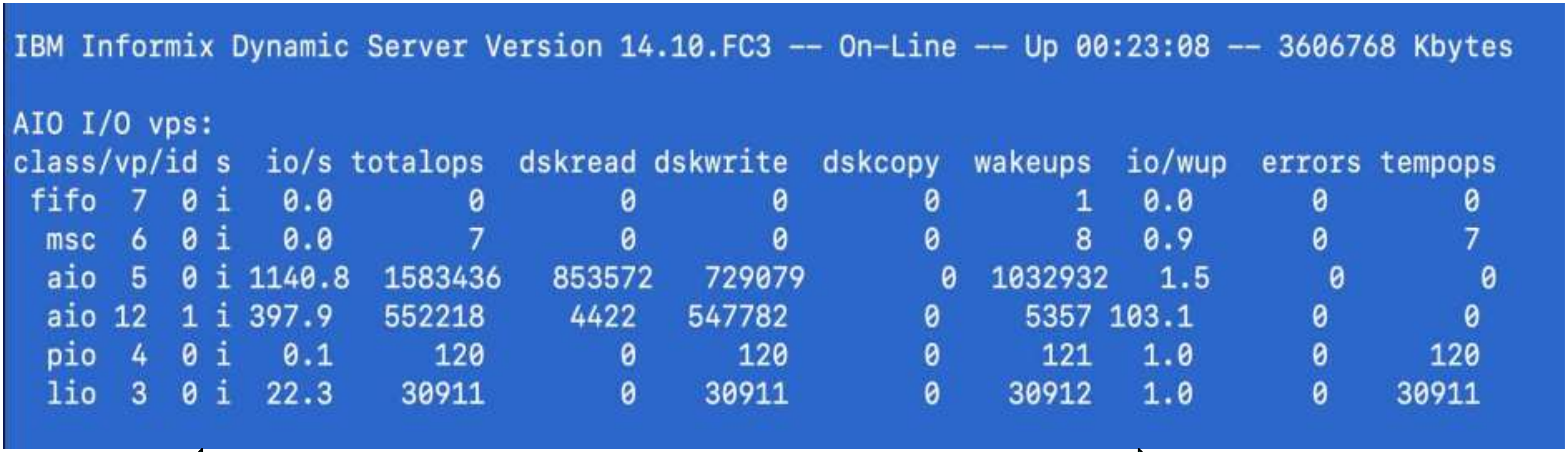


```
IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:19:57 -- 3606768 Kbyte

AIO global files:
gfd pathname         bytes read     page reads  bytes write    page writes io/s
3   rootdbs          3080192        1504        623329280      304360      719.7
        op type     count          avg. time
        seeks       0              N/A
        reads       1466           0.0000
        writes      6612           0.0017
        kaio_reads  0              N/A
        kaio_writes 0              N/A

4   logdbs           0              0           3341301760     1631495     2462.7
        op type     count          avg. time
        seeks       0              N/A
        reads       0              N/A
        writes      28402          0.0004
        kaio_reads  0              N/A
        kaio_writes 0              N/A

5   datadbs          1568690176     765962      2223282176     1085587     72124.2
        op type     count          avg. time
        seeks       0              N/A
        reads       761698         0.0000
        writes      68087          0.0001
        kaio_reads  0              N/A
        kaio_writes 0              N/A

6   tmpdbs           0              0           8192           4           188786.3
        op type     count          avg. time
        seeks       0              N/A
        reads       0              N/A
        writes      2              0.0000
        kaio_reads  0              N/A
        kaio_writes 0              N/A
```

Shows no KAIO

Monitor > 0.01

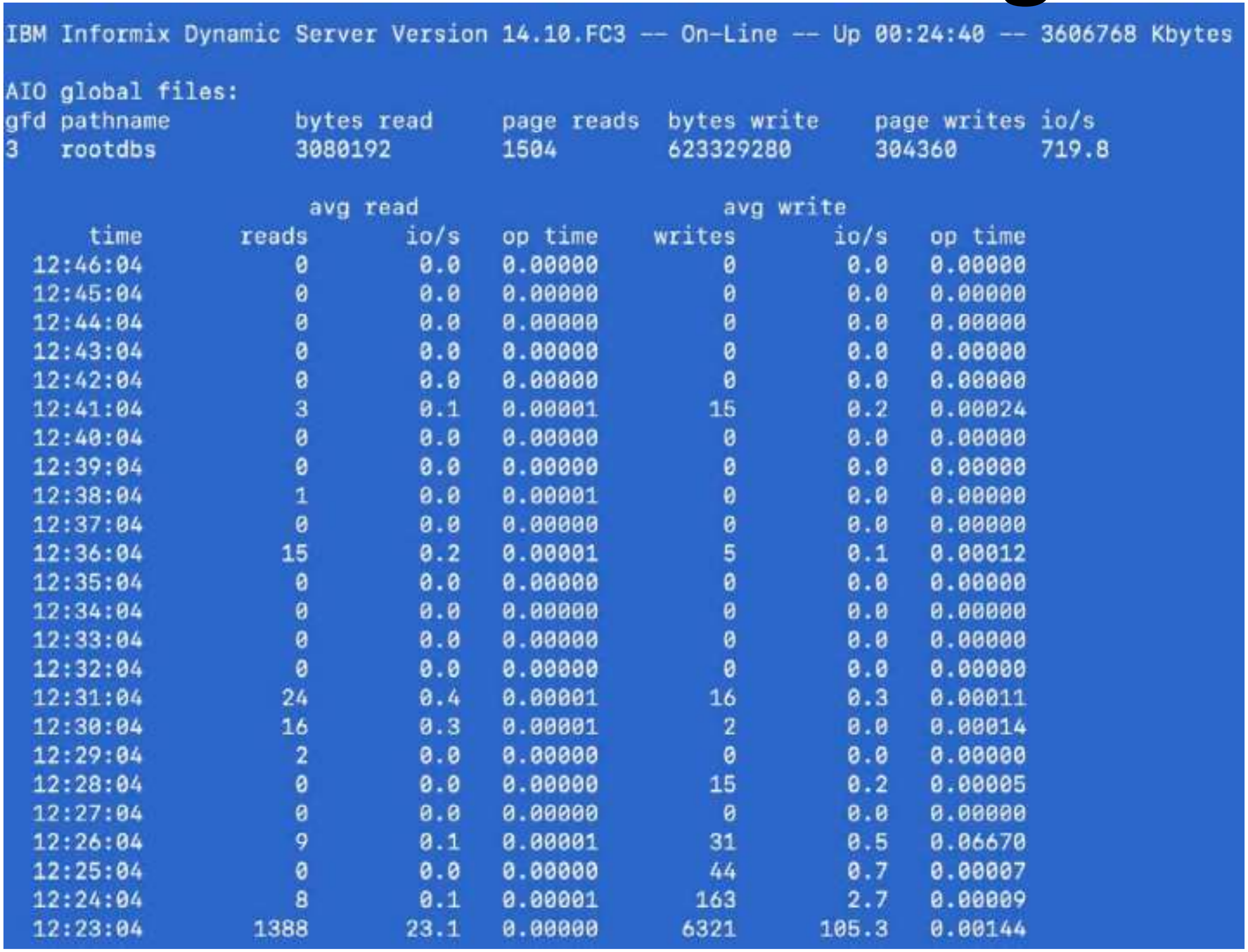
# Monitor with onstat –g iov



```
IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:23:08 -- 3606768 Kbytes

AIO I/O vps:
class/vp/id s  io/s totalops  dskread dskwrite  dskcopy  wakeups  io/wup  errors tempops
 fifo  7  0 i   0.0        0        0        0        0        1   0.0       0       0
  msc  6  0 i   0.0        7        0        0        0        8   0.9       0       7
  aio  5  0 i 1140.8  1583436   853572   729079        0  1032932   1.5       0       0
  aio 12  1 i 397.9   552218     4422   547782        0     5357 103.1       0       0
  pio  4  0 i   0.1      120        0      120        0      121   1.0       0     120
  lio  3  0 i  22.3    30911        0    30911        0    30912   1.0       0   30911
```

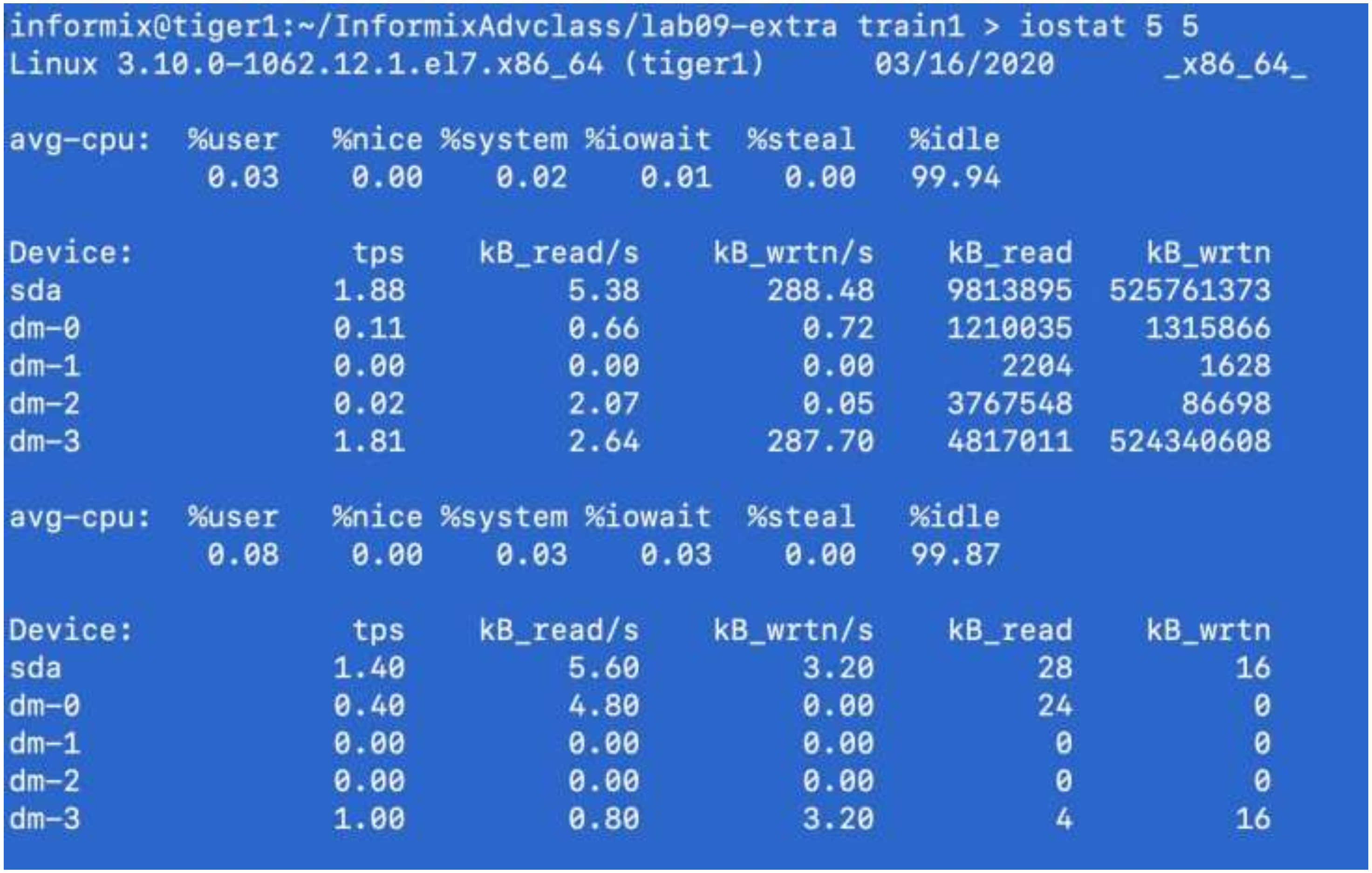
Is the IO balance among AIO VPs?

# Monitor with onstat –g ioh



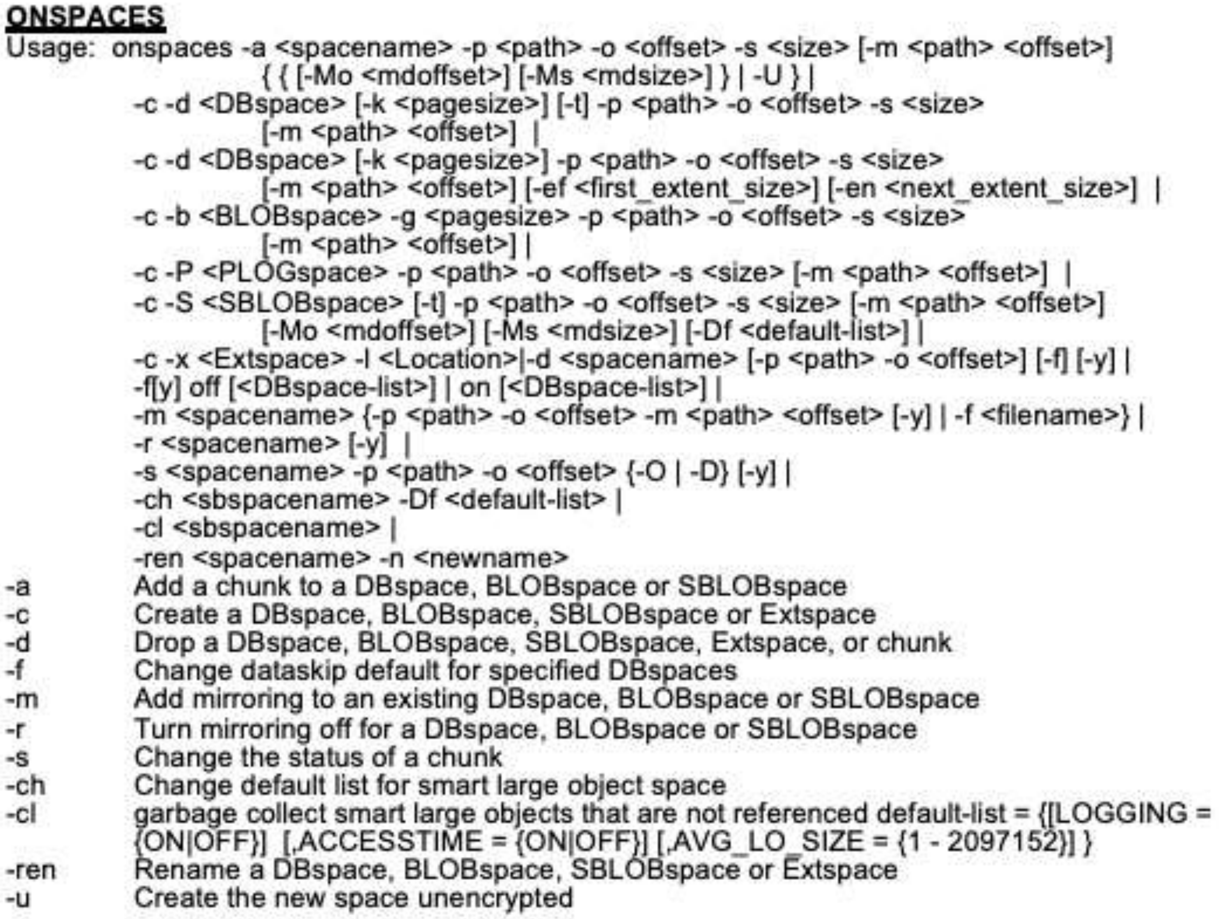
```
IBM Informix Dynamic Server Version 14.10.FC3 -- On-Line -- Up 00:24:40 -- 3606768 Kbytes

AIO global files:
gfd pathname          bytes read      page reads  bytes write     page writes io/s
3   rootdbs           3080192         1504        623329280       304360      719.8

                        avg read                              avg write
      time         reads          io/s    op time    writes          io/s   op time
  12:46:04             0           0.0    0.00000         0           0.0   0.00000
  12:45:04             0           0.0    0.00000         0           0.0   0.00000
  12:44:04             0           0.0    0.00000         0           0.0   0.00000
  12:43:04             0           0.0    0.00000         0           0.0   0.00000
  12:42:04             0           0.0    0.00000         0           0.0   0.00000
  12:41:04             3           0.1    0.00001        15           0.2   0.00024
  12:40:04             0           0.0    0.00000         0           0.0   0.00000
  12:39:04             0           0.0    0.00000         0           0.0   0.00000
  12:38:04             1           0.0    0.00001         0           0.0   0.00000
  12:37:04             0           0.0    0.00000         0           0.0   0.00000
  12:36:04            15           0.2    0.00001         5           0.1   0.00012
  12:35:04             0           0.0    0.00000         0           0.0   0.00000
  12:34:04             0           0.0    0.00000         0           0.0   0.00000
  12:33:04             0           0.0    0.00000         0           0.0   0.00000
  12:32:04             0           0.0    0.00000         0           0.0   0.00000
  12:31:04            24           0.4    0.00001        16           0.3   0.00011
  12:30:04            16           0.3    0.00001         2           0.0   0.00014
  12:29:04             2           0.0    0.00000         0           0.0   0.00000
  12:28:04             0           0.0    0.00000        15           0.2   0.00005
  12:27:04             0           0.0    0.00000         0           0.0   0.00000
  12:26:04             9           0.1    0.00001        31           0.5   0.06670
  12:25:04             0           0.0    0.00000        44           0.7   0.00007
  12:24:04             8           0.1    0.00001       163           2.7   0.00009
  12:23:04          1388          23.1    0.00000      6321         105.3   0.00144
```

# Monitor Unix Disk IO with iostat



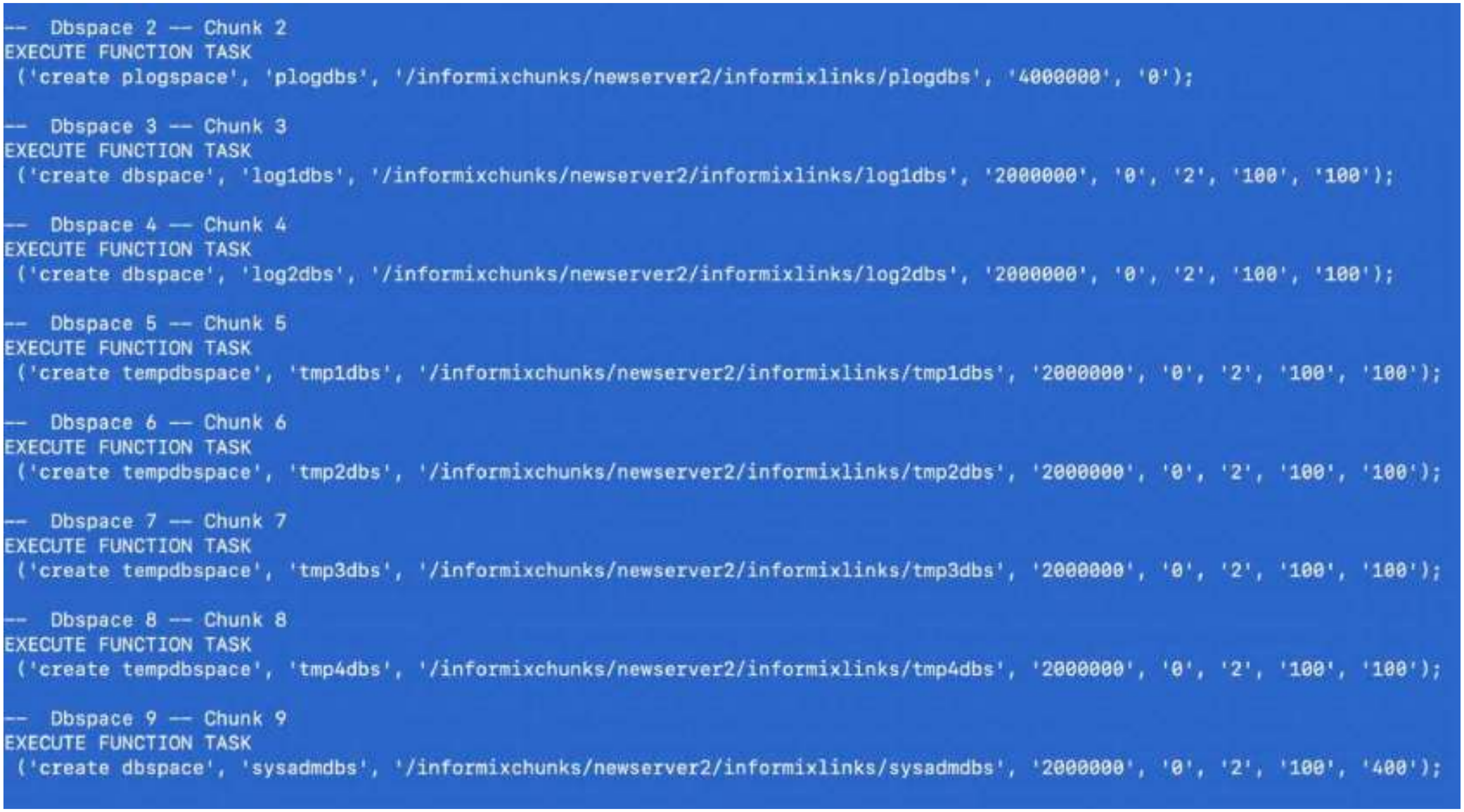
```
informix@tiger1:~/InformixAdvclass/lab09-extra train1 > iostat 5 5
Linux 3.10.0-1062.12.1.el7.x86_64 (tiger1)      03/16/2020      _x86_64_

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.03    0.00    0.02    0.01    0.00   99.94

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               1.88         5.38       288.48    9813895  525761373
dm-0              0.11         0.66         0.72    1210035    1315866
dm-1              0.00         0.00         0.00       2204       1628
dm-2              0.02         2.07         0.05    3767548      86698
dm-3              1.81         2.64       287.70    4817011  524340608

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.08    0.00    0.03    0.03    0.00   99.87

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               1.40         5.60         3.20         28         16
dm-0              0.40         4.80         0.00         24          0
dm-1              0.00         0.00         0.00          0          0
dm-2              0.00         0.00         0.00          0          0
dm-3              1.00         0.80         3.20          4         16
```

# The onspaces Command



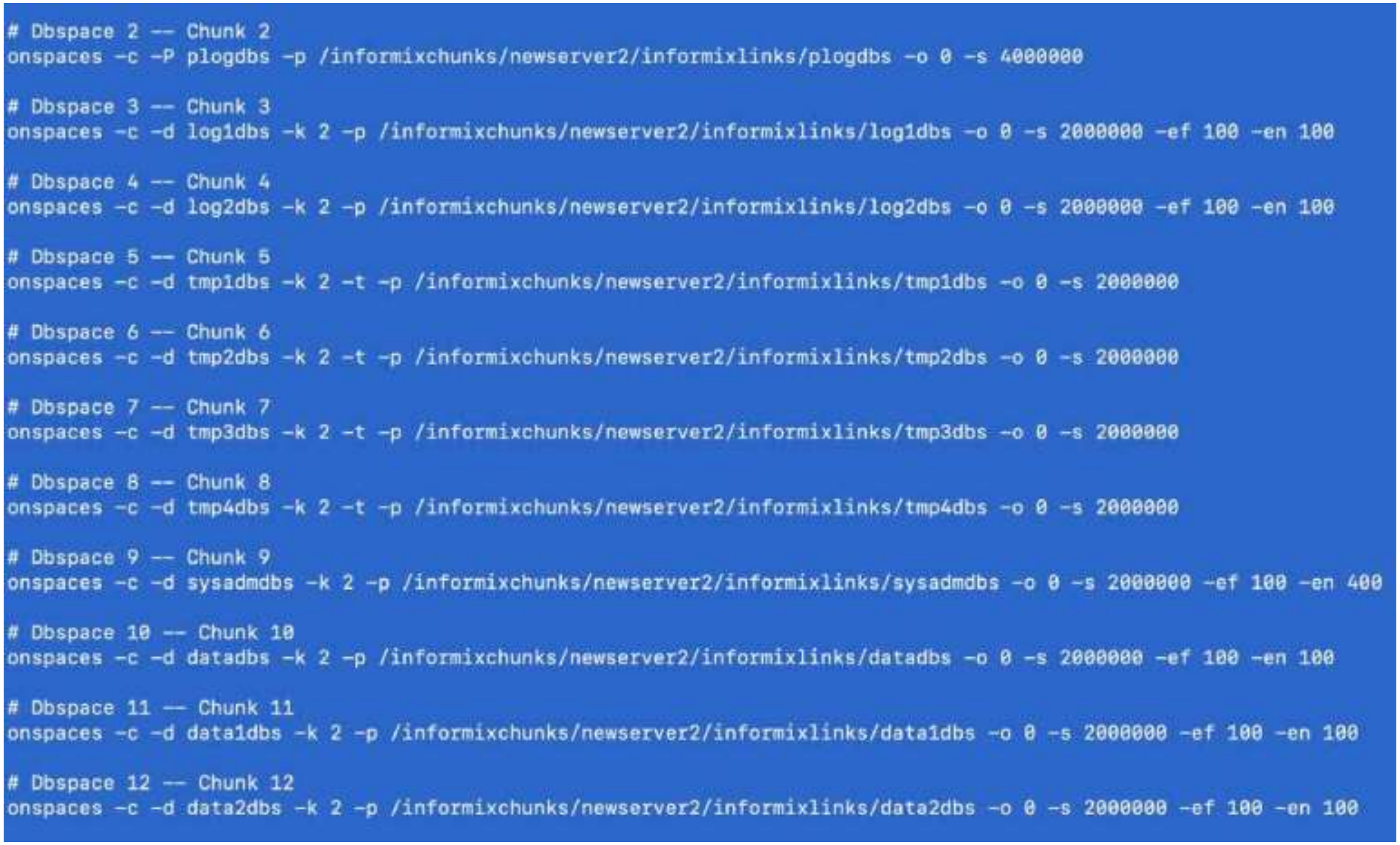
```
ONSPACES
Usage:  onspaces -a <spacename> -p <path> -o <offset> -s <size> [-m <path> <offset>]
                          { { [-Mo <mdoffset>] [-Ms <mdsize>] } | -U } |
             -c -d <DBspace> [-k <pagesize>] [-t] -p <path> -o <offset> -s <size>
                          [-m <path> <offset>]  |
             -c -d <DBspace> [-k <pagesize>] -p <path> -o <offset> -s <size>
                          [-m <path> <offset>] [-ef <first_extent_size>] [-en <next_extent_size>]  |
             -c -b <BLOBspace> -g <pagesize> -p <path> -o <offset> -s <size>
                          [-m <path> <offset>] |
             -c -P <PLOGspace> -p <path> -o <offset> -s <size> [-m <path> <offset>]  |
             -c -S <SBLOBspace> [-t] -p <path> -o <offset> -s <size> [-m <path> <offset>]
                          [-Mo <mdoffset>] [-Ms <mdsize>] [-Df <default-list>] |
             -c -x <Extspace> -l <Location>|-d <spacename> [-p <path> -o <offset>] [-f] [-y] |
             -f[y] off [<DBspace-list>] | on [<DBspace-list>] |
             -m <spacename> {-p <path> -o <offset> -m <path> <offset> [-y] | -f <filename>} |
             -r <spacename> [-y]  |
             -s <spacename> -p <path> -o <offset> {-O | -D} [-y] |
             -ch <sbspacename> -Df <default-list> |
             -cl <sbspacename> |
             -ren <spacename> -n <newname>
-a           Add a chunk to a DBspace, BLOBspace or SBLOBspace
-c           Create a DBspace, BLOBspace, SBLOBspace or Extspace
-d           Drop a DBspace, BLOBspace, SBLOBspace, Extspace, or chunk
-f           Change dataskip default for specified DBspaces
-m           Add mirroring to an existing DBspace, BLOBspace or SBLOBspace
-r           Turn mirroring off for a DBspace, BLOBspace or SBLOBspace
-s           Change the status of a chunk
-ch          Change default list for smart large object space
-cl          garbage collect smart large objects that are not referenced default-list = {[LOGGING =
             {ON|OFF}]  [,ACCESSTIME = {ON|OFF}] [,AVG_LO_SIZE = {1 - 2097152}] }
-ren         Rename a DBspace, BLOBspace, SBLOBspace or Extspace
-u           Create the new space unencrypted
```

# Using Sysadmin Task



```
--  Dbspace 2 -- Chunk 2
EXECUTE FUNCTION TASK
 ('create plogspace', 'plogdbs', '/informixchunks/newserver2/informixlinks/plogdbs', '4000000', '0');

--  Dbspace 3 -- Chunk 3
EXECUTE FUNCTION TASK
 ('create dbspace', 'log1dbs', '/informixchunks/newserver2/informixlinks/log1dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 4 -- Chunk 4
EXECUTE FUNCTION TASK
 ('create dbspace', 'log2dbs', '/informixchunks/newserver2/informixlinks/log2dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 5 -- Chunk 5
EXECUTE FUNCTION TASK
 ('create tempdbspace', 'tmp1dbs', '/informixchunks/newserver2/informixlinks/tmp1dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 6 -- Chunk 6
EXECUTE FUNCTION TASK
 ('create tempdbspace', 'tmp2dbs', '/informixchunks/newserver2/informixlinks/tmp2dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 7 -- Chunk 7
EXECUTE FUNCTION TASK
 ('create tempdbspace', 'tmp3dbs', '/informixchunks/newserver2/informixlinks/tmp3dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 8 -- Chunk 8
EXECUTE FUNCTION TASK
 ('create tempdbspace', 'tmp4dbs', '/informixchunks/newserver2/informixlinks/tmp4dbs', '2000000', '0', '2', '100', '100');

--  Dbspace 9 -- Chunk 9
EXECUTE FUNCTION TASK
 ('create dbspace', 'sysadmdbs', '/informixchunks/newserver2/informixlinks/sysadmdbs', '2000000', '0', '2', '100', '400');
```

# Using onspaces



```
# Dbspace 2 -- Chunk 2
onspaces -c -P plogdbs -p /informixchunks/newserver2/informixlinks/plogdbs -o 0 -s 4000000

# Dbspace 3 -- Chunk 3
onspaces -c -d log1dbs -k 2 -p /informixchunks/newserver2/informixlinks/log1dbs -o 0 -s 2000000 -ef 100 -en 100

# Dbspace 4 -- Chunk 4
onspaces -c -d log2dbs -k 2 -p /informixchunks/newserver2/informixlinks/log2dbs -o 0 -s 2000000 -ef 100 -en 100

# Dbspace 5 -- Chunk 5
onspaces -c -d tmp1dbs -k 2 -t -p /informixchunks/newserver2/informixlinks/tmp1dbs -o 0 -s 2000000

# Dbspace 6 -- Chunk 6
onspaces -c -d tmp2dbs -k 2 -t -p /informixchunks/newserver2/informixlinks/tmp2dbs -o 0 -s 2000000

# Dbspace 7 -- Chunk 7
onspaces -c -d tmp3dbs -k 2 -t -p /informixchunks/newserver2/informixlinks/tmp3dbs -o 0 -s 2000000

# Dbspace 8 -- Chunk 8
onspaces -c -d tmp4dbs -k 2 -t -p /informixchunks/newserver2/informixlinks/tmp4dbs -o 0 -s 2000000

# Dbspace 9 -- Chunk 9
onspaces -c -d sysadmdbs -k 2 -p /informixchunks/newserver2/informixlinks/sysadmdbs -o 0 -s 2000000 -ef 100 -en 400

# Dbspace 10 -- Chunk 10
onspaces -c -d datadbs -k 2 -p /informixchunks/newserver2/informixlinks/datadbs -o 0 -s 2000000 -ef 100 -en 100

# Dbspace 11 -- Chunk 11
onspaces -c -d data1dbs -k 2 -p /informixchunks/newserver2/informixlinks/data1dbs -o 0 -s 2000000 -ef 100 -en 100

# Dbspace 12 -- Chunk 12
onspaces -c -d data2dbs -k 2 -p /informixchunks/newserver2/informixlinks/data2dbs -o 0 -s 2000000 -ef 100 -en 100
```